VOL. 5 NO. 12
DECEMBER 1983

neat notes

THE NEWSLETTER OF NEW ENGLAND APPLE TREE

December 1983 N.E.A.T. Meeting

The December meeting will be held at 7:30 P.M. on the third Wednesday of the month (as usual), December 21, 1983 at the Mitre cafeteria. Mitre is on rte 62 in Bedford Ma. Take route 3 north from rte 128 to the first exit (rte 62).

December's MEETING

Our Speaker for the December meeting is Patrick O'Neil, President of Exec Software, located in Lexington Mass. Pat will come here to tell us about Apple communications in general, and about his company's exciting new communications package, TermExec, in particular (look for the article starting on page 5). It should be an informative meeting, so be sure to be there!

THE FUTURE SCHEDULE IS TENTATIVELY AS FOLLOWS:

December 21,1983------EXEC SOFTWARE

January 18,1984-------??????????

February 15, 1984-----PARKER BROS GAMES!

March 21, 1984--------ELF: THE RESOURCE MANAGER

Computer Express

Have you been looking for a reliable, convenient, low-cost computer outlet?

Your search has ended!

Computer Express is here. Located just around the corner in Needham, MA.

We offer super low prices and back them up with great service.

VISA

Call (617) 449-5484 for orders and information

MasterCard

neat notes

THE NEWSLETTER OF NEW ENGLAND APPLE TREE

NEAT NOTES STAFF

Editor/Publisher.....
John Shaw,643-2679
Editor/Beginners Corner
Jack Sheehan,924-1197
Editor/Writer-Assembly Language
John Shaw,643-2679
Games Reviewer.....
Mark Hessman,475-2617
Advertising Manager....
HELP !!!

N.E.A.T. OFFICERS and CHAIRMEN

President..Terry Scheller,244-4833
Vice-pres...Jack Thompson,655-1896
Vice-pres....Elliot Entin,862-6418
Secretary....Alice Janian,623-6757
Treasurer.....David Lynn,655-6164
Library.....
Input.....M.Gooding,264-4237
Sales.....D.McMullin,444-4384
DOM (at D.Goldstein,273-0567
meeting).E.Mac Lauchlan,878-2229
Outside User Group Contact.....
Larry Starnes,263-1795
Membership.Joe Stivaletta,861-1899
Auditor.....Don Foster
Greeter.....Jack Thompson,253-8688

SPECIAL INTEREST GROUPS

SIGED.....Jack Sheehan,924-1197
SIGCP/M....Joe Stivaletta,861-1899
SIGASSY.....John Shaw,643-2679
SIGCOMM.....
SIGPASCAL..Jerry Connelly,497-9858
SIGNEW.....Lou Tarsa,547-3289

Copyright 1982, New England Apple Tree, All rights reserved.

All original Articles may be reprinted without permission by computer club newsletters if they are not copyrighted by the author. Reprints must clearly indicate the author's name, the name and address of NEAT, a copyright notice and the date of the issue. Two copies of the document in which the reprint appears should also be sent to:
Attn. Editor,NEAT Notes
P.O.Box 2652
Woburn,Ma 01881

DECEMBER HIGHTLIGHTS

NEAT Meeting Notes
.....Page 4
Calendar
by Raines Cohen.....Page 4
The Editor Speaketh
.....Page 4
DECEMBER SPEAKER: EXEC SOFTWARE
.....Page 5
BANK STREET WRITER: A Review
by Guy Desloovere.....Page 7
Machine Language Spoken Here
by Dr. J. .....Page 9
KOALAPAD: A Review
by Earl H. Benser.....Page 13
HTABbing In 80 Columns
by John Wrenholt.....Page 14
PRODOS: A New Operating System
by John Wrenholt.....Page 16
Membership Application and
DOM Order Blank.....Page 17

N.E.A.T. MEETING MINUTES
November 16, 1983

Present: John Shaw, Dave Lynn, Joe Stivaletta, Elliot Entin, Mike Gooding and Alice Janian.

Elliot called the meeting to order at 7:30 P.M.

Meeting highlights: SWAP NIGHT This is the one meeting of the year where the club's officers stand before the general members and swap tall tales about how they shot an elephant in their pajamas, how they were instrumental in the discovery of the Salk vaccine, and how they spirited Anastasia out of Russia in 1917 (of course, only Elliot is old enough to have much credibility on this last one!). Terry didn't have any good stories to swap, so he missed the meeting altogether.

After the swapping of tall tales, Elliot and John brought out the Wesson oil, spray cheese, and Saran Wrap; everyone had a good time thereafter.

Officer Muldoon broke up the meeting at 9:00 P.M.

CALENDAR
Compiled by Raines Cohen

December 17 9:30AM
Apple Guild
Massasoit Community College
call 767-1303

December 21 7:30PM
New England Apple Tree
Mitre Cafeteria
call 497-9050

December 24 2:30PM
Newton Apple Kids
Location: ?????
call 965-0033

January 4 7:00PM
New England Computer Society
Mitre Cafeteria
call 646-3080

January 18 7:30PM
New England Apple Tree
Mitre Cafeteria
call 497-9050

January 21 9:30AM
Apple Guild
Massasoit Community College
call 767-1303

January 28 2:30PM
Newton Apple Kids
Location: ?????
call 965-0033

February 1 7:00PM
New England Computer Society
Mitre Cafeteria
call 646-3080

THE EDITOR SPEAKETH

Brief message this month: if all goes well, you will not receive NEAT NOTES in January. 'How's that?', you ask. Well, yours truly is planning to head off to India in the latter part of Decmeber, and won't be back till the middle of January. However, if you or anyone you know would like to take a crack at putting together January's issue, I'll be glad to help you get started; give me a call if you're interested. Otherwise, au revoir until Febuary, and a Happy Holidays to you all!

J.J.S.

DECEMBER SPEAKER: EXEC SOFTWARE

Okay, so you bought a modem for your Apple and are all set to start on that long-awaited dream of 'networking' with the rest of the world. Fine. The problem is, your Apple does not come with communications software. Okay, you say to yourself, 'I'll go pick up a communications package to match my new modem'. Great. Which one? Consider this: most of the rest of the world will want to talk to you using 80 characters per line. Sometimes the world has something so great to say that you will want to save it on disk (for posterity). Sometimes the world has nothing great to say and you won't want to save it at all. And sometimes, just sometimes, the world will have something to say, but you won't know whether or not you want to save it until you have seen the entire message scroll by - an 'after the fact' save, so to speak.

Fortunately, these and other pressing communications issues will all be cleared up at our December NEAT meeting, when the good folks from Exec Software drop by to give us a demonstration of their new communications package: TermExec.

'Great', you must be saying to yourself, 'there are already over a dozen terminal packages for the Apple. Why do I want to know about this one?' Well, consider TermExec's many features:

A Simple Command System: For many purposes, such as manipulating files on your Apple's disks, commands are easier to use than menus. TermExec commands are patterned after the Apple DOS commands, but will allow abbreviations, and will prompt you if you type an incomplete command. At all times, TermExec places a complete HELP facilitiy at your disposal to give you a list of the actions you have available.

Unattended Long File Capture and Send: TermExec will capture a file up to the full capacity of your diskette. Moreover, the file will be received as an integral whole, not broken up into several pieces. TermExec also supports the Christensen XMODEM protocol to eliminate errors in file transmission.

Up to 1200 Baud Operation: This is the highest speed that telephone lines will allow.

Exclusive "Backscrolling" Feature: TermExec will let you review text that has rolled off the top of your screen; up to 7,000 characters can be recovered.

Edited File Capture, After the Fact: You can go back and capture to disk exactly what you want from the 7,000 character Scrolling Buffer, and leave out the "boiler plate" around the important information.

80 Column Support: The Apple][e 80 column board is fully supported - even at 1200 baud. TermExec will also emulate 80 columns on the 40 column Apple][+ using horizontal scrolling (ala VISICALC) inside the 40 column window. This feature will let you check document right margins and table layout (not to mention graphs).

Auto Answer and Dial Up: You can place TermExec into Terminal Answer mode, so that it will respond to commands sent over the

telephone just as if you typed them in yourself.

EXEC Commands: An EXEC file contains an arbitrary sequence of commands which you have created. Using TermExec's EXEC command, you can invoke the entire sequence of commands inside an EXEC file. If that weren't enough, you can also provide arguments to your EXEC command, and TermExec will place them in the proper positions in your sequence of commands.

Automatic Operation: The EXEC facility can be combined with the flexible command language of TermExec. As a simple example: you can create and issue an EXEC command to wait some specified period of time (for reduced late night time charges), then connect through an auto-dial model to a remote Data service, issue a series of requests, capture the results, and log out. If you wish, the EXEC command can be written to do this several times in one night with different services. Furthermore, TermExec does not require a clock to perform these tasks since it uses the underlying clock in the Apple's 6502.

Macro Commands: With TermExec, you can define short sequences of commands and save them in memory for repeated use during your terminal session.

File Copy and Convert: TermExec can make a copy of any file on the disks in your Apple, and can perform file conversions, from Applesoft, Integer and Binary to Text and back, to permit the unencumbered transfer of program files.

TermExec also has a full screen editor (complete with provisions to insert/delete text, search for key words, etc.), and can simultaneously send to your printer the same information you see scrolling by on your screen.

TermExec is provided on an unprotected disk: you have no worries about making backups, and can move TermExec to a hard disk or to a disk emulator board for faster access. In addition, this makes it possible for Exec Software to provide automatic upgrade DOS patches through their telephone support bulletin board (see below).

TermExec sells for $79.95; for an additional $19.95, the good folks at Exec Software will sign you up for membership in the DELPHI information service (a normal membership sells for $49.95). Exec Software maintains a telephone HOTLINE and a 300 baud bulletin board which serves as an online newsletter to their customers. Through this bulletin board, the floks at Exec can announce new features for TermExec, answer complex questions, and download new general purpose EXEC files.

TermExec will work with an Apple][e or an Apple][+, with 48K of memory, at least one disk, and the following modems: Hayes Micromodem, Novation Applecat, Hayes Smartmodem, The Operator, and acoustic couplers connected through the Apple Super Serial card, the CCS Card, and the SSM Card.

If you cannot make the December meeting, you can call the folks at Exec Software directly: (617) 862-3170.

We are looking forward to seeing you at the December meeting.

BANK STREET WRITER

by Broderbund Software

Reviewed for NEAT
by Guy Desloovere

Bank Street Writer is a word processing program which is fairly powerful, yet very easy and even fun to use, and also quite inexpensive (about $50). It is nearly impossible to bomb Bank Street Writer, as the ESCape key will extricate you beningly from just about any wrong move. The program is locked, but you get a free backup disk in the original package. Not only that, but Broderbund guarantees free replacements forever. The disk will boot on either II/II+ or IIe and adjust automatically to the different keyboards. If you have 64K on your II+ or 128K on your IIe, Bank Street Writer will automatically use the extra memory. The following reflects my experience with my 64K II+.

The back of the disk contains a very clear tutorial which covers the most important aspects of the program. By simply running this tutorial, you would be able to use the program after 10 minutes without ever reading the 28-page booklet. However, you should read the booklet anyway. It is very easy to read and explains some refinements which are not covered in the tutorial or in the screen prompts. It also describes the utility programs. Utility #1 allows you to adjust Bank Street Writer to your hardware configuration. The default configuration assumes a single disk drive, which is perfectly adequate. Note that Bank Street Writer includes its own lowercase generator, but is compatible with most lowercase chips and shift-key modifications. In the default configuration, Bank Street Writer uses SHIFT-N to capitalize the next letter and double SHIFT-N to lock into uppercase. Utility #1 also sets some of the printing parameters such as left, top and bottom margins, and the default values of other parameters, which can be changed at printing time. Utility #2 displays the (optional) passwords you used to save files, in case you forgot, while Utility #3 converts between Bank Street Writer binary files and standard Apple text files.

When the main program is running, the screen always shows 4 prompt lines at the top that tell you which mode you are in, which commands are active in this mode, and how to change mode. Below the 4 prompt lines is a framed rectangular area into which 18 scrolling lines of 38 characters of text can be entered in free form.

There are 3 modes: Write, Edit and Transfer. Entering text, even corrections, is always done in Write mode. Text is entered at the cursor. When making additions, you can insert an infinite number of characters and the text to the right of the cursor will be shifted right, with word wrap to the next lines. You can even insert a whole file from disk. Conversely, when making deletions, you use the arrows to delete characters and the text at right will be shifted left, again with word wrap. With a 64K system, the document in memory can be up to 3200 words long. Pressing CTRL-S will tell you how much space you have left. The ESCape key is used as a toggle to move between Write mode and Edit mode.

Edit mode is used primarily to

move the cursor with the cursor keys, i.e. I, J, K and M for the II+, or with keys B (to beginning), E (to end), U (up 12 lines), and D (down 12 lines). Note that to make a correction anywhere except right at the cursor, you must first go to Edit mode to position the cursor, and then go back to Write mode to enter new characters and/or delete old ones. This takes a bit of getting used to. In Edit mode, you can also ERASE or MOVE a block of up to 15 lines. If you change your mind, you can UNERASE or MOVEBACK the latest block. You can also FIND or REPLACE a string of up to 29 characters. These commands are selected by using the arrows and the space bar to step through the corresponding prompts, while the cursor is used to delimit the block or string. A minor complaint is that you can MOVE only forward. The TRANSFER MENU command takes you from Edit mode to Transfer mode. Returning to Edit mode is controlled by the ubiquitous ESCape key.

Transfer mode is used to transfer to/from disk or to printer. You can RETRIEVE, SAVE, DELETE, RENAME, PRINT-DRAFT or PRINT-FINAL, and also CLEAR memory. File names cannot be longer than 8 characters, which can be a bit limiting. PRINT-DRAFT gives you an exact copy of what you see on the screen, i.e. 38 characters per line. This makes it easier to edit from a marked-up copy, but I prefer to work from a PRINT-FINAL copy which is the only way of knowing what the final document will look like. It is not possible to display the final format on the screen, which makes it difficult to position a date in the top right corner of the page, and practically impossible to use hyphenation to improve justification. On the other hand, you can request Bank Street Writer to display page breaks and you can adjust them within limits. You can also center a title on the printed page by pressing CTRL-C.

When the PRINT-FINAL command is invoked, a series of questions are displayed about printing parameters, together with the default values, which were set with Utility #1. Questions cover number of characters per line, spacing between lines, whether pages are to be numbered or not, whether at top or bottom, and what to use for page heading, if anything. Accepting the default values obviously speeds up the process, which can otherwise be a bit tedious. The printing parameter values are valid for the whole document and cannot be changed locally. Another question you are asked is whether the file to be printed is a continuation of a previous file, so that if you had to save a long text in separate sections, you can still print it as a single document, with the same page heading and continuing page numbering. Conversely, another question allows you to PRINT parts only of a file. The same is true of SAVE.

In summary, Bank Street Writer does not do everything, but it does quite a lot, and what it does, it does easily and well. Before Bank Street Writer, I used to look for excuses not to write. Now, I look for opportunities to use Bank Street Writer. However slowly I peck at the keyboard, and however many mistakes or changes I make, it gives me a clean document with a minimum of effort. I can heartily recommend Bank Street Writer.

MACHINE LANGUAGE SPOKEN HERE

By Dr. J

The doctor is in.

Hello there gang! In October, I began our little series on using Applesoft strings from assembly language. Back then, I briefly introduced the 'dynamic memory allocation' method Apple uses to store strings. This month, we are going to reinforce your understanding of this method with a clever little exercise: locating a substring inside a larger string using a short machine language routine. All set to join me? Good; let's press on.

You may recall our brief discussion of Applesoft's dynamic memory allocation method for strings, to wit: every string has a three-byte descriptor which acts like a 'forwarding address' to tell the Apple (1) where the value of the string is stored in memory, and (2) the length of the string. For example, suppose the three byte descriptor for a particular string is stored beginning at memory location $871:

$871: 05 E0 95

This descriptor indicates that the first character in the string can be found at the address $95E0, and the string has a length of 5 bytes.

A string's descriptor can be found using the Applesoft ROM routine PTRGET. Thus, if we wish to perform some operation on a particular string, we need merely use PTRGET to locate where in memory the Apple has stored that string's descriptor, look up the descriptor and find the length of the string and where its value is stored in memory. Sound simple enough? Good; let's see how we can use this information to develop a routine to locate a substring inside a larger string.

Suppose we wanted to find out if the string:

B$ = "IS THE TIME"

appears in its entirety inside another string:

A$ = "NOW IS THE TIME FOR"

[In a more practical application, the first string might represent, say, a person's name while the second string might represent a large continuous record containing client names.]

How can we determine if a match exists? Well, first consider this: B$ has a length of 11 bytes, while A$ has a length of 19 bytes: at most 8 bytes in A$ can fail to match those bytes in B$. Put another way: suppose we begin our search for a match by starting at the <u>first</u> byte in A$ and ask whether the next 11 bytes perfectly match the bytes in B$. If the test succeeds we can say that the match was found at the first byte in A$. If the test fails we repeat the test, this time starting at the <u>second</u> byte in A$. If that test succeeds we say the match was found at the second byte in A$, otherwise we pick up the test again at the third byte, and so on. Now, it shouldn't be too hard to see that no match can possibly be found if we haven't established a match by the time we pick up our test starting at the <u>tenth</u> byte in A$.

Clearly, we can establish some fairly straightforward rules for conducting a search for a match. Let LA represent the length of A$, and LD represent the length of B$. Then, a match cannot possibly exist if either of the following conditions are true:

LA = 0 (A$ is null)

LB = 0 (B$ is null)

LB > LA

Furthermore, if we haven't established a match by the time we pick up our test beginning at byte:

LA + 2 - LB

in A$, then a match cannot possibly be found.

Each 'test', as it were, begins by setting a pointer, PTR, to some value (initially, it will be set to 1), and asking if the next LB bytes in A$, starting from the byte at position PTR and ending at the byte at position PTR+LB, perfectly match the bytes in B$. If they do, we quit. Otherwise, we increment PTR by one. If PTR equals LA + 2 - LB, we quit; otherwise we begin the test anew.

Our machine language routine to establish a match between two strings is shown in Listing #1. This routine accepts two strings as arguments; we arbitrarily refer to these strings as A$ and B$, although they can be called by any names inside our Applesoft program.

```
SOURCE FILE: SUBSTRING COMPARE

0000:       1 ;*************************************
0000:       2 ;*                                   *
0000:       3 ;*  AMPER SUBSTRING SEARCH           *
0000:       4 ;*     BY LEE REYNOLDS               *
0000:       5 ;*                                   *
0000:       6 ;* TO USE:                           *
0000:       7 ;*    %A$,B$                         *
0000:       8 ;*    X = PEEK(26)                   *
0000:       9 ;*  REM X = POSITION OF              *
0000:      10 ;*      1ST CHARACTER IN             *
0000:      11 ;*      COMMON IS B$ IS A            *
0000:      12 ;*      SUBSTRING OF A$              *
0000:      13 ;*                                   *
0000:      14 ;*  CALL -A.P.P.L.E. JAN 81          *
0000:      15 ;*                                   *
0000:      16 ;*************************************
0000:      17 ;
0000:      18 ; SET UP THE EQUATES:
0000:      19 ;
0006:      20 STRLO   EQU  $06      ; LOW ADDRESS FOR STRING
0007:      21 STRHI   EQU  $07      ; HIGH ADDRESS FOR STRING
0008:      22 SUBLO   EQU  $08      ; LOW ADDRESS FOR SUBSTRING
0009:      23 SUBHI   EQU  $09      ; HIGH ADDRESS FOR SAME
0018:      24 STRLEN  EQU  $18      ; STORAGE FOR STRING LENGTH
0019:      25 SUBLEN  EQU  $19      ; STORAGE FOR UNDSTRING LENGTH
001A:      26 PTR     EQU  $1A      ; POINTER TO POSN IN STRING
001B:      27 LSTPOS  EQU  $1B      ; LAST TESTING POSITION
0000:      28 ;
0000:      29 ; APPLESOFT ROM ROUTINES
0000:      30 ;
DEBE:      31 CHKCOM  EQU  $DEBE
DFE3:      32 PTRGET  EQU  $DFE3
0000:      33 ;
0000:      34 ; START THE PROGRAM
0000:      35 ;
----- NEXT OBJECT FILE NAME IS SUBSTRING COMPARE.OBJ0
0300:      36          ORG  $300
0300:20 E3 DF 37          JSR  PTRGET    ; GET THESTRING POINTER
0303:85 06    38          STA  STRLO     ; AND STORE THE LOW/HIGH
0305:84 07    39          STY  STRHI     ; ADDRESS OF SAME
```

Lines 37-55 locate the descriptor to A$ and pull out those bytes giving the length of the string (lines 42, 43), and the address where the string's value is stored in memory (lines 48-55). Being cheap, we like to conserve storage space, so we use the memory locations STRLO/STRHI for two purposes: first to hold the location of the string's descriptor, and the second to hold the address of the string's location in memory.

Lines 57-79 repeat the steps just described for B$.

The guts of our routine begins at line 83. We begin by checking for the first three conditions we noted above and quit if any one is true. If none of the three are true, we establish the last position we can begin a test in A$ (lines 89-92).

Lines 93-113 define the test for a match. If a match fails inside a test (lines 95-96) we exit the test, increment our pointer, PTR, check to see it does not exceed our last position value, and repeat the test anew. If PTR exceeds our last position counter

```
0307:A0 00      40          LDY  #$0          ; CLEAR THE POINTER
0309:84 1A      41          STY  PTR          ; GET THE LENGTH OF THE STRING
030B:B1 06      42          LDA  (STRLO),Y
030D:85 18      43          STA  STRLEN       ; AND SAVE IT
030F:           44 ;
030F:           45 ; LET'S NOW GET THE ADDRESS TO THE ACTUAL
030F:           46 ; LOCATION OF THE STRING
030F:           47 ;
030F:C8         48          INY
0310:B1 06      49          LDA  (STRLO),Y    ; GET THE LOW BYTE
0312:48         50          PHA               ; AND STORE TEMPORARILY
0313:C8         51          INY
0314:B1 06      52          LDA  (STRLO),Y    ; GET THE HIGH BYTE
0316:85 07      53          STA  STRHI        ; AND SAVE IT
0318:68         54          PLA               ; RECOVER THE LOW BYTE
0319:85 06      55          STA  STRLO        ; AND SAVE IT AS WELL.
031B:20 BE DE   56          JSR  CHKCOM       ; GET THE COMMA
031E:20 E3 DF   57          JSR  PTRGET       ; GET THESUBSTRING POINTER
0321:85 08      58          STA  SUBLO        ; STORE THE DESCRIPTOR
0323:84 09      59          STY  SUBHI
0325:           60 ;
0325:           61 ; NOW, RECOVER THE SUBSTRING'S LENGTH AND THE
0325:           62 ; ADDRESS TO ITS ACTUAL VALUE, JUST AS WE DID
0325:           63 ; FOR THE STRING:
0325:           64 ;
0325:A0 00      65          LDY  #$0
0327:B1 08      66          LDA  (SUBLO),Y    ; GET THE LENGTH
0329:85 19      67          STA  SUBLEN       ; AND SAVE IT
032B:           68 ;
032B:           69 ; LET'S NOW GET THE ADDRESS TO THE ACTUAL
032B:           70 ; LOCATION OF THE SUBSTRING
032B:           71 ;
032B:C8         72          INY
032C:B1 08      73          LDA  (SUBLO),Y    ; GET THE LOW BYTE
032E:48         74          PHA               ; AND PUSH ON THE STACK FOR TEMP. STORAGE
032F:C8         75          INY
0330:B1 08      76          LDA  (SUBLO),Y    ; GET THE HIGH BYTE
0332:85 09      77          STA  SUBHI        ; AND SAVE IT
0334:68         78          PLA               ; RECOVER THE LOW BYTE
0335:85 08      79          STA  SUBLO        ; AND SAVE IT AS WELL.
```

(lines 108-109), we exit the routine. Finally, if a match is found we return the position in A$ where the match succeeded in memory location 26 (hex $1A).

This, in a nutshell, demonstrates one of the more valuable utilities for manipulating strings in assembly language. In February, we will discover how to extend this utility to a far grander scale: finding a match between a single string and an entire array of strings!

Till then, good luck!

The doctor is out.

LISTING #2

```
100 POKE 1013,76: POKE 1014,0: POKE 1015,3: REM  JMP $300
110 PRINT
120 INPUT "PLEASE ENTER YOUR STRING ";YY$
140 PRINT : INPUT "PLEASE ENTER YOUR SUBSTRING ";XX$
160 A YY$,XX$
180 PRINT : PRINT "A MATCH WAS FOUND IN POSITION "; PEEK (26)
200 END
```

```
0337:                   80 ;
0337:                   81 ; FINISHED WITH THE PRELIMINARIES
0337:                   82 ;
0337:                   83 ENTER  LDA  STRLEN   ; STRING LENGTH
0337:A5 18              84        BEQ  EXIT     ; IF NULL STRING, LEAVE
0339:F0 1E              85        CMP  SUBLEN   ; COMPARE TO SUBSTRING LENGTH
033B:C5 19              86        BCC  EXIT     ; IF SUBLEN > STRLEN THEN EXIT
033D:90 1A              87        LDA  SUBLEN
033F:A5 19              88        BEQ  EXIT     ; EXIT IF SUBSTRING IS NULL
0341:F0 16              89        LDA  STRLEN   ; SET STRING LENGTH
0343:A5 18              90        SEC
0345:38                 91        SBC  SUBLEN
0346:E5 19              92        STA  LSTPOS   ; LAST POSN TO TEST
0348:85 18              93 TEST   LDY  #0
034A:A0 00              94 LOOP   LDA  (STRLO),Y ; COMPARE CHAR IN STRING
034C:B1 06              95        CMP  (SUBLO),Y ; TO CHAR IN SUBSTRING
034E:D1 08              96        BNE  NXTTST   ; NEXT TEST IF NO MATCH
0350:D0 08              97        INY           ; POINT TO NEXT CHAR IN STRING
0352:C8                 98        CPY  SUBLEN   ; COMPARE TO SUBSTRING LENGTH
0353:C4 19              99        BCC  LOOP     ; IF LESS, THEN CONTINUE TEST
0355:90 F5             100 ;
0357:                  101 ; WE'VE FOUND A MATCH. RETURN WITH PTR + 1
0357:                  102 ;
0357:                  103       INC  PTR
0359:60                104 EXIT  RTS
035A:E6 1A             105 NXTTST INC  PTR      ; UPDATE PTR TO THE BEGINNING
035C:                  106 ; POSITION OF THE NEXT SEARCH
035C:A5 12             107       LDA  LSTPOS   ; GET THE LAST POSN TO BEGIN TEST
035E:C5 1A             108       CMP  PTR      ; COMPARE
0360:90 09             109       BCC  NOFIND   ; IF LESS, NO SUBSTRING FOUND
0362:E6 06             110       INC  STRLO    ; UPDATE ADDRESS FOR NEXT
0364:D0 02             111       BNE  NEXT     ; TEST TO NEXT CHR POSN.
0366:E6 07             112       INC  STRHI
0368:4C AA 03          113 NEXT  JMP  TEST
036B:A9 00             114 NOFIND LDA  #$0      ; STRING NOT FOUND
036D:A7 00             115       STA  PTR      ; RETURN THIS VALUE TO THE USER
036F:85 1A             116       RTS           ; BYE BYE
0371:60                117 EXIT  RTS
```

KOALAPAD: A REVIEW

by Earl H. Benser

[Reprinted from the Aug '83 MINI'APP'LES Newsletter]

In the July issue of Mini'app'les, there was a comment that the Koalapad was the greatest graphics tool since sliced bread. I'm not a major graphics nut, but for a long time I have been trying to figure a way to get the Apple Graphics Tablet (FCC to the contrary). The price, even at cost, however, was more than my wallet would bear. So when I saw this first response to the Koalapad, I had to at least give it a look. Some look... I watched one in action for about five minutes and then bought it. That was the beginnings of a rough evening. It was about 2 A.M. before I got too tired to continue playing with it.

In a nutshell, the Koalapad is a 4 inch or so square touch sensitive surface which translate the position of a special stylus (or your finger) into position codes for input into the game port. It is essentially a 'solid state' joystick. That's nice, but with the software that comes with the Koalapad, you can do amazing graphic details.

When the software disc boots, it places a graphics menu on the screen. (Please use a color TV or monitor, b&w just doesn't do it justice.) The items on the menu are selected by using the Koalapad to position a displayed cursor over the item of your choice, and pressing the left control button (PB0?). The menu provides various point and line drawing options, with various "brushes", to generate straight and freeform lines, options to generate square and circular frames, with and without filled interiors, an option to fill the interior of a drawn irregular area, and both "erase" and "store/load" options. Graphics can be done on the normal full screen, or on a 7X magnified screen. Two sets of colors are provided, with nine colors each, which can be used for any selected draw or fill mode. In the "erase" option, the erased screen can be left solid with any of the 18 colors available (actually 16, since black and white are common to both sets).

For the most part, the Koalapad just lets you do your thing, the way you want to do it. Within a few minutes of starting, the operation of the Koalapad and the menu become almost second nature. And it's all too easy to start dreaming of becoming a digital Rembrandt. (Now you also begin to wish your Apple had superdetailed graphics capability.) One of the first lessons learned appears to be that certain color options don't necessarily go well side by side. Strange colors can begin to show up at the boundaries due to Apple's color coding techniques. Also, when you want to fill areas on the screen, the first color works, but trying to overfill with a second color works sometimes, and sometimes not. There appears to be a priority sequence that I haven't quite figured out yet. (Maybe I should take the time to read the instructions?)

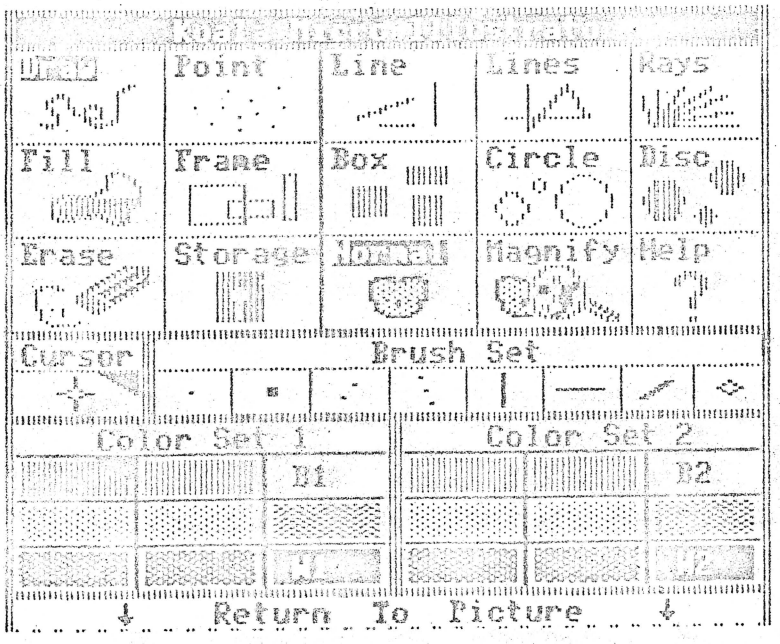
Speaking of instructions, they are brief, but effective, and not really necessary. It's very easy to get the hang of the functions just from playing with the

Koalapad, and everyone will do that.

The Koalapad will work with any graphics image available on disk, no matter what the source. It can be used for titling, but I would suggest using another program, such as the Beagle Bros. version for text work. With the Koalapad in the 'magnify' mode, it takes very steady nerves to get the letters aligned and proper.

The Koalapad lists for $125, with the possibility of a somewhat lower net price. Either way, I think that it's one heck of a good deal just for the graphics capability. In addition, it supposedly can also work as a joystick for other programs. I tried that with less than total success, but that just might be me rather than the Koalapad.

All in all, the Koalapad is the best thing since sliced bread, as the Mini'app'les note said. If you've ever had a desire to try doing any graphics work, and even if you haven't had a desire, you'll get one as soon as you see the Koalapad in action!



HTABBING IN 80 COLUMNS

by John Wrenholt

[Reprinted from
The Scarlett Letter]

When I first purchased my Apple IIe, one of the first "bugs" that I discovered was that the Applesoft command HTAB didn't work properly in the eighty-column mode. If you have an Apple IIe, you can see what the problem is by running this short program:

```
10 TEXT:HOME
20 PRINT CHR$(4)"PR#3":
   REM TURN ON THE 80 COLUMN
      BOARD
30 HTAB 10 :
   PRINT"THIS HTAB DOESN'T
   WORK."
40 HTAB 10 :
   PRINT"BUT THIS ONE DOES,"
50 HTAB 50 :
   PRINT"AND THIS WILL NEVER
   WORK"
```

I checked the manual for the eighty-column text card and sure enough, someone from Apple had also noticed the problem. While the manual did not actually call the problem a bug, it did suggest a solution. Use POKE 1403,X instead of HTAB X whenever you are working with the 80-column card. That sounded like a reasonable solution and a little testing showed that it did indeed work. Now that I had a solution to the problem, I suppose that I should have considered the case closed, but I just couldn't help but wonder: "what does memory location 1403 have to do with the 80-column card?"

A quick look at the memory map for the Apple IIe showed that

memory location 1403 was right in the middle of the text screen area! Now I was really curious. After some snooping around in the Apple reference Manual, I found the answer. The manual states: "Each peripheral card which uses a slot in the Apple is allocated eight bytes of memory. These 56 bytes (8 bytes * 7 slots) of main memory are located in the RAM memory reserved for text and low-resolution graphics displays, but these particular locations are not displayed on the screen. Programs in ROM on peripheral cards are able to use these locations for temporary data storage." In other places in the manual, I found these memory locations referred to as either "scratch pad memory", or "screen holes".

What all this means is that each peripheral card slot in an Apple has 8 bytes of RAM that it can use without worrying about memory conflicts with other software. Figure 1 shows that you can find the location of any screen hole by taking the Base Address in the table and adding the slot number. This makes it possible for peripheral cards to run in any slot. In machine language, a subroutine can directly reference any one of the eight screen holes by loading the slot number into the Y-register and executing LDA $0478,Y, where $0478 (decimal 1144) is the desired base address.

As you can see from Figure 1, memory location 1403 is reserved for use with slot number 3. The 80-column text card uses this memory location to store its current cursor position. The problem I pointed out earlier occurs because Applesoft's HTAB is unaware of the 80-column text card, and continues to update the normal horizontal cursor position.

FIGURE 1: APPLE SCREEN HOLES

BASE ADDR	SLOT NUMBER 1	2	3	4	5	6	7
1144	1145	1146	1147	1148	1149	1150	1151
1272	1273	1274	1275	1276	1277	1278	1279
1400	1401	1402	1403	1404	1405	1406	1407
1528	1529	1530	1531	1532	1533	1534	1535
1656	1657	1658	1659	1660	1661	1662	1663
1784	1785	1786	1787	1788	1789	1790	1791
1912	1913	1914	1915	1916	1917	1918	1919
2040	2041	2042	2043	2044	2045	2046	2047

I decided to write an ampersand (&) HTAB routine which would work with either the 40 or 80-column displays. The program turned out to be rather simple as you can see from the listing. As written, it will work only with the Apple IIe 80-column text card, but it can easily be adapted to work with other 80-column cards, such as the Videx.

The following Applesoft listing demonstrates how you can use &HTAB in your own programs.

```
10 HOME: TEXT
20 POKE 1013,76: POKE 1014,0:
   POKE 1015,3: REM SET UP
   AMPERSAND
30 PRINT CHR$(4)"PR#3":
   REM TURN ON 80-COLUMN BOARD
40 &HTAB 30: PRINT "IT WORKS"
50 &HTAB 30: PRINT "IT WORKS"
60 FOR X = 1 TO 70
70 &HTAB X: PRINT "IT WORKS"
80 NEXT
90 END
```

```
1000 *--------------------------------
1010 *      &HTAB FOR APPLE ][e
1020 *      STARTED FEBRUARY 1983
1030 *--------------------------------
1040 HTAB.POS             .EQ $057B
1050 COL80.STATUS         .EQ $C01F
1060 SYNCHR               .EQ $DEC0
1070 ILLEGAL.QTY.ERR      .EQ $E199
1080 GETBYT               .EQ $E6F8
1090 AS.HTAB              .EQ $F7E7
1100         .OR $300
1110 START
1120         LDA #150     TOKEN FOR HTAB?
1130         JSR SYNCHR   NO.. THEN SYNTAX ERROR
1140         JSR GETBYT   GET VALUE INTO X-REG
1150         LDA COL80.STATUS ARE WE IN 80-COLUMN MODE?
1160         BMI .2       YES... THEN .2
1170         JMP AS.HTAB+3 NO.. THEN LET HTAB HANDLE IT
1180 .2      DEX
1190         CPX #80      IS X¶80?
1200         BCS .3       YES... THEN ERROR
1210 .1      STX HTAB.POS NO...STORE X IN 1403
1220         RTS          EXIT
1230 .3      JMP ILLEGAL.QTY.ERR
```

****
PRODOS: A NEW OPERATING SYSTEM
****

by John Wrenholt

Reprinted from THE SCARLETT LETTER
August 1983, Vol. 2 No. 7

Apple will soon begin shipping a new Disk Operating System for the Apple II and Apple IIe family of computers. ProDos, which stands for Professional Disk Operation System, is an entirely new system not just an upgrade of DOS 3.3.

Although many of the commands remain the same, such as BRUN and OPEN, DOS 3.3 users will have much to learn about the new system. Those who are familar with SOS, the Apple ///'s operating system, should feel right at home with ProDos since it uses a hierarchical file structure identical to that of SOS's. It has also copied the file format of SOS, so that data and media can be directly exchanged between an Apple II and an Apple ///.

Anyone, who can remeber the process of MUFFINing their programs from DOS 3.2 to DOS 3.3 is probably not all that thrilled to find out that once again they are going to have to spend long hours transferring and converting their programs to a new operating system. But before you send your Apple flying towards the trashcan, let me point out two items which may ease your anger slightly.

First, the change from DOS 3.3 to ProDos is done entirely in the software. There will be no PROMs to switch on the disk controller card. This means that DOS 3.3 and ProDos can live in harmony with each other. If a program works in the DOS 3.3 enviroment, you won't be forced to transfer it to ProDos.

Secondly, ProDos offers several improvements over DOS 3.3. Here are just some of the advantages of ProDos:

1. It supports any disk which uses Apple protocols without changes in the software. This means that applications written for the Apple Drive will also operate on a hard disk system.

2. It accesses the disk about 8 times faster than DOS 3.3. A 16K BLOAD will take only 3 seconds with ProDos compared to 17 seconds under DOS.

3. Apple IIe owners with 128K memory will be able to use the extra 64K as a RAM disk.

There are some disadvantages to ProDos, however. First it requires a 64K Apple II. If you only have 48K of memory, you will be unable to use ProDos until you purchase a language card. Also, ProDos does not currently support Integer Basic. This means that Integer Basic programs will not run in the ProDos enviroment.

The following DOS 3.3 Commands are not supported by ProDos: INIT; INT; FP; MAXFILES; MON; NOMON; and VERIFY. The following new commands have been added: CAT (40 column catalog); CHAIN; CREATE; FLUSH; PREFIX; RESTORE; STORE; - (dash, a smart run command); and CATALOG (80 column catalog). I found - to be especially nice to use. It can be used in place of RUN, BRUN, and EXEC. It will automatically determine the file typpe and act accordingly.

This Command Interpreter does allow some new features to be added to Applesoft. For example, "garbage collection" is now handled by the ProDos CI rather than by Applesoft; in a few seconds rather than 2-15 minutes! The CI also makes it possible for you to add your own commands to Applesoft.

So whatever you do, don't throw your Apple away yet. I know you're gonna like ProDos!

MEMBERSHIP APPLICATION

1. Name _____ Date _____
2. Street _____
3. City _____ State _____ Zip _____
4. Phones (a) Home _____ (b) Work _____
5. Software background _____
6. Hardware background _____
7. What will you use your APPLE for? _____
8. In what area can the club assist you? _____
9. Are you available to assist other members? _____
10. Total amount enclosed _____

1983 Membership Dues are

UNITED STATES..... $ 15.00

CANADA..... $ 21.00

(OFFICE USE ONLY)

PAID BY: cash check RECEIPT #_____ STATUS: New Renewal MEM #_____

-DISKETTE ORDER FORM-

Diskettes are sold to members and non-members and can be picked up at the meeting or mailed anywhere in the continental U.S. Circle desired selection.

Games	I	II	III	IV	V	VI	VII					
Education	I	II		Math	I	Scientific		I	Applesoft Tutor			
Pascal	I	II	III	IV	V	VI	Diversi-DOS			ABBS		Hayes Util.
D.O.M.1981	J	F	M	A	M	J	J	A	S	O	N	D
D.O.M.1982	J	F	M	A	M	J	J	A	S	O	N	D
D.O.M.1983	J	F	M	A	M	J	J	A	S	O	N	D

All D.O.M.'s up to the current month $5.00 picked up, $6.50 mailed. The current D.O.M. is copied free on your initialized diskette at the meeting.

All catagorized diskettes include a description of the programs and are available for $5.00 if picked up and $6.50 if they must be mailed.

Non-members prices are $10.00 picked up and $11.50 mailed.

Check here if you want diskettes mailed ( ).

NAME _____

ADDRESS _____

CITY/STATE _____

ZIP _____

MEMBERSHIP CARD # _____

Make check payable to N.E.A.T.

Send to:

N.E.A.T.-Attn: Librarian
P.O.Box 2652
Woburn, MA 01801

Total enclosed $ _____.____

neat
New England Apple Tree
Box 2652, Woburn, MA 01801

BULK RATE
U.S. POSTAGE
PAID
NEWTONVILLE, MA. 02160
PERMIT NO. 57931

EXPIRES 02/84

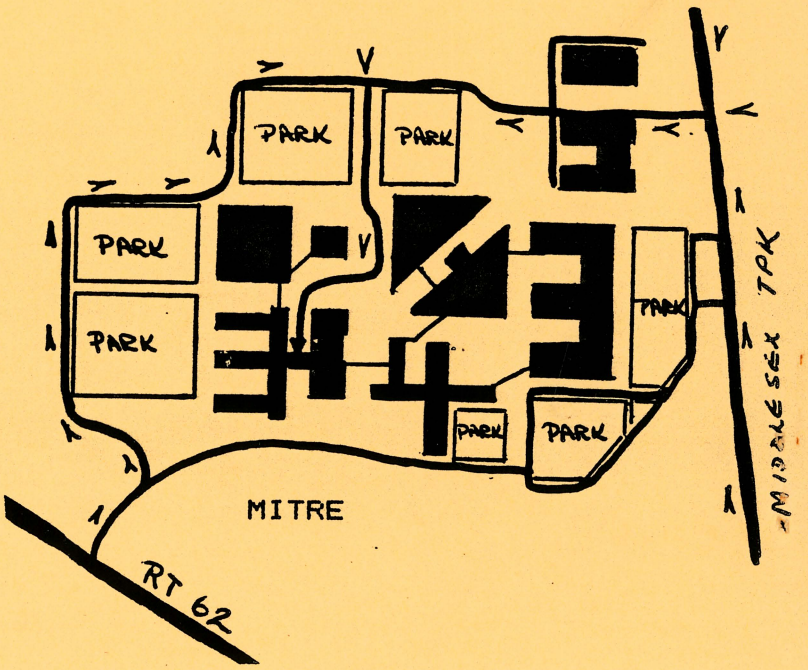
ADVERTISING RATES

Single Issue
- Full Page.....$40.00
- Half Page.....$20.00
- Quarter Page.....$10.00

3 Month series or Longer
- Full Page.....$30.00
- Half Page.....$15.00
- Quarter Page.....$7.50

Inserts letter size accepted at the same rates, subject to weight limitations.

Personal advertisements of members, e.g., swap notices, personal articles for sale, etc. accepted without charge.

Please send ad copy in black and white (half-tones are permited but no bleeds) by the last Wednesday of the month preceeding the month of issue to:

N.E.A.T. Notes
Attention: Editor
P.O.Box 2652
Woburn, MA. 01801